Contents

Upgrade with Helm

6 minute read page test

Follow this guide to upgrade and configure an Istio mesh using Helm for in-depth evaluation. This guide assumes you have already performed an installation with Helm for a previous minor or patch version of Istio.

The Helm charts used in this guide are the same

underlying charts used when installing Istio via Istioctl or the Operator.

This feature is currently considered alpha.

Prior to Istio 1.9.0, installations using the Helm charts required hub and tag arguments: `--set global.hub="docker.io/istio"` and `--set global.tag="1.8.2"`. As of Istio 1.9.0 these are no longer required.

Prerequisites

1. Download the Istio release.
2. Perform any necessary platform-specific setup.
3. Check the Requirements for Pods and Services.
4. Install a Helm client with a version higher than 3.1.1.

Helm 2 is not supported for installing Istio.

The commands in this guide use the Helm charts that

are included in the Istio release package located at `manifests/charts`.

Upgrade steps

Change directory to the root of the release package and then follow the instructions below.

> The default chart configuration uses the secure third party tokens for the service

account token projections used by Istio proxies to authenticate with the Istio control plane. Before proceeding to install any of the charts below, you should verify if third party tokens are enabled in your cluster by following the steps describe here. If third party tokens are not enabled, you should add the option `--set global.jwtPolicy=first-party-jwt` to the Helm install commands. If the `jwtPolicy` is not set correctly, pods associated with `istiod`, gateways or workloads with injected Envoy proxies will not get deployed due to the missing `istio-token` volume.

Before upgrading Istio, it is recommended to run the `istioctl x precheck` command to make sure the upgrade is compatible with your environment.

```
$ istioctl x precheck
✔ No issues found when checking the cluster. Istio is safe to in
stall or upgrade!
To get started, check out https://istio.io/latest/docs/setup/get
ting-started/
```

Helm does not upgrade or delete CRDs when performing an upgrade. Because of this restriction, an additional step is required

when upgrading Istio with Helm.

Create a backup

Before upgrading Istio in your cluster, we recommend creating a backup of your custom configurations, and restoring it from backup if necessary:

```
$ kubectl get istio-io --all-namespaces -oyaml > "$HOME"/istio_resource_backup.yaml
```

You can restore your custom configuration like this:

```
$ kubectl apply -f "$HOME"/istio_resource_backup.yaml
```

Canary upgrade (recommended)

You can install a canary version of Istio control plane to validate that the new version is compatible with your existing configuration and data plane using the steps below:

Note that when you install a canary version of the `istiod` service, the underlying cluster-wide resources from the base chart are shared across your primary and canary installations.

Currently, the support for canary upgrades for Istio ingress and egress gateways is actively in development and is considered `experimental`.

1. Upgrade the Kubernetes custom resource

definitions (CRDs):

```
$ kubectl apply -f manifests/charts/base/crds
```

2. Install a canary version of the Istio discovery chart by setting the revision value:

```
$ helm install istiod-canary manifests/charts/istio-control
/istio-discovery \
    --set revision=canary \
    -n istio-system
```

3. Verify that you have two versions of `istiod` installed in your cluster:

```
$ kubectl get pods -l app=istiod -L istio.io/rev -n istio-s
ystem
  NAME                                READY   STATUS    RESTART
S   AGE   REV
  istiod-5649c48ddc-dlkh8             1/1     Running   0
    71m   default
  istiod-canary-9cc9fd96f-jpc7n       1/1     Running   0
    34m   canary
```

4. Follow the steps here to test or migrate existing workloads to use the canary control plane.
5. Once you have verified and migrated your workloads to use the canary control plane, you can uninstall your old control plane:

```
$ helm delete istiod -n istio-system
```

6. Upgrade the Istio base chart:

```
$ helm upgrade istio-base manifests/charts/base -n istio-sy
stem --skip-crds
```

Stable revision labels (experimental)

Stable revision labels are only supported when updating Istio from and to Istio versions 1.10+.

Manually relabeling namespaces when moving them to a new revision can be tedious and error-prone. Revision tags solve this problem. Revision tags are stable identifiers that point to revisions and can be used to avoid relabeling namespaces. Rather than relabeling the namespace, a mesh operator can simply change the tag to point to a new revision. All namespaces labeled with that tag will be updated at the same time.

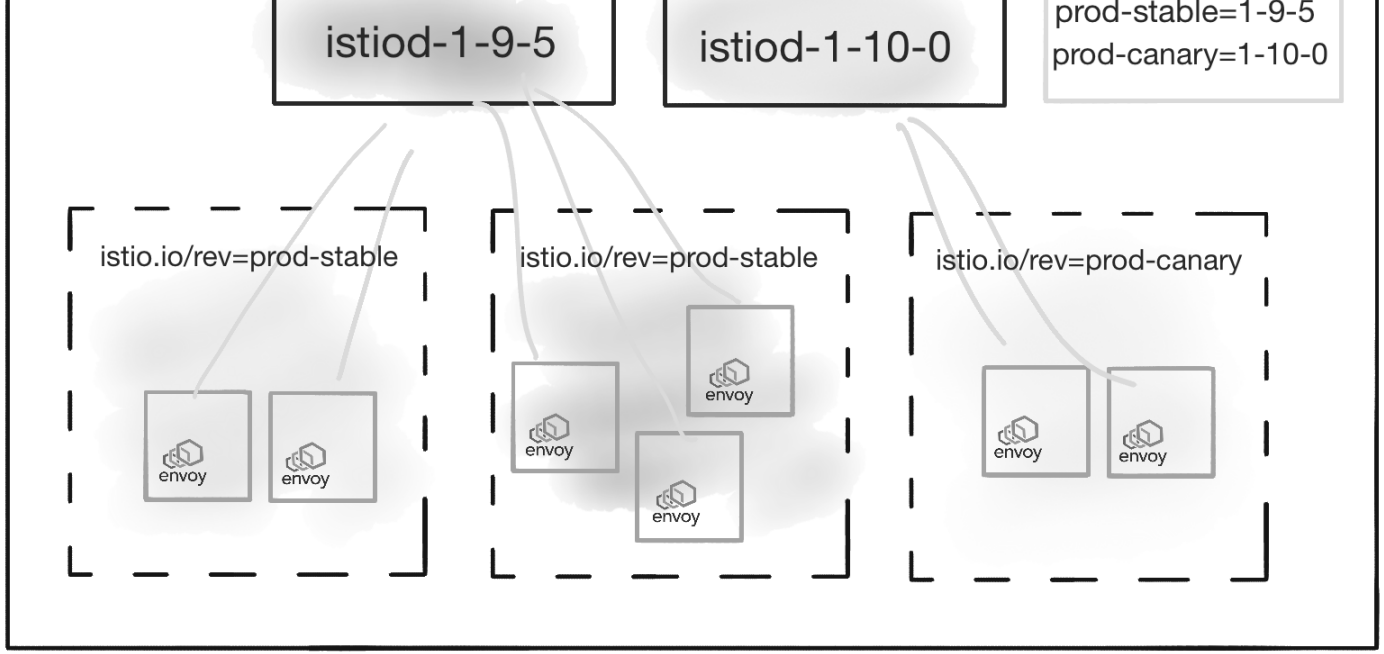
Usage

Consider a cluster with two revisions installed, `1-9-5` and `1-10-0`. The cluster operator creates a revision tag `prod-stable`, pointed at the older, stable `1-9-5` version, and a revision tag `prod-canary` pointed at the newer `1-10-0` revision. That state could be reached via these commands:

```
$ helm template istiod manifests/charts/istio-control/istio-discovery -s templates/revision-tags.yaml --set revisionTags={prod-stable} --set revision=1-9-5 -n istio-system | kubectl apply -f -
$ helm template istiod manifests/charts/istio-control/istio-discovery -s templates/revision-tags.yaml --set revisionTags={prod-canary} --set revision=1-10-0 -n istio-system | kubectl apply -f -
```

These commands create new `MutatingWebhookConfiguration` resources in your cluster, however, they are not owned by any Helm chart due to `kubectl` manually applying the templates. See the instructions below to uninstall revision tags.

The resulting mapping between revisions, tags, and namespaces is as shown below:



Two namespaces pointed to prod-stable and one pointed to prod-canary

The cluster operator can view this mapping in addition to tagged namespaces through the `istioctl`

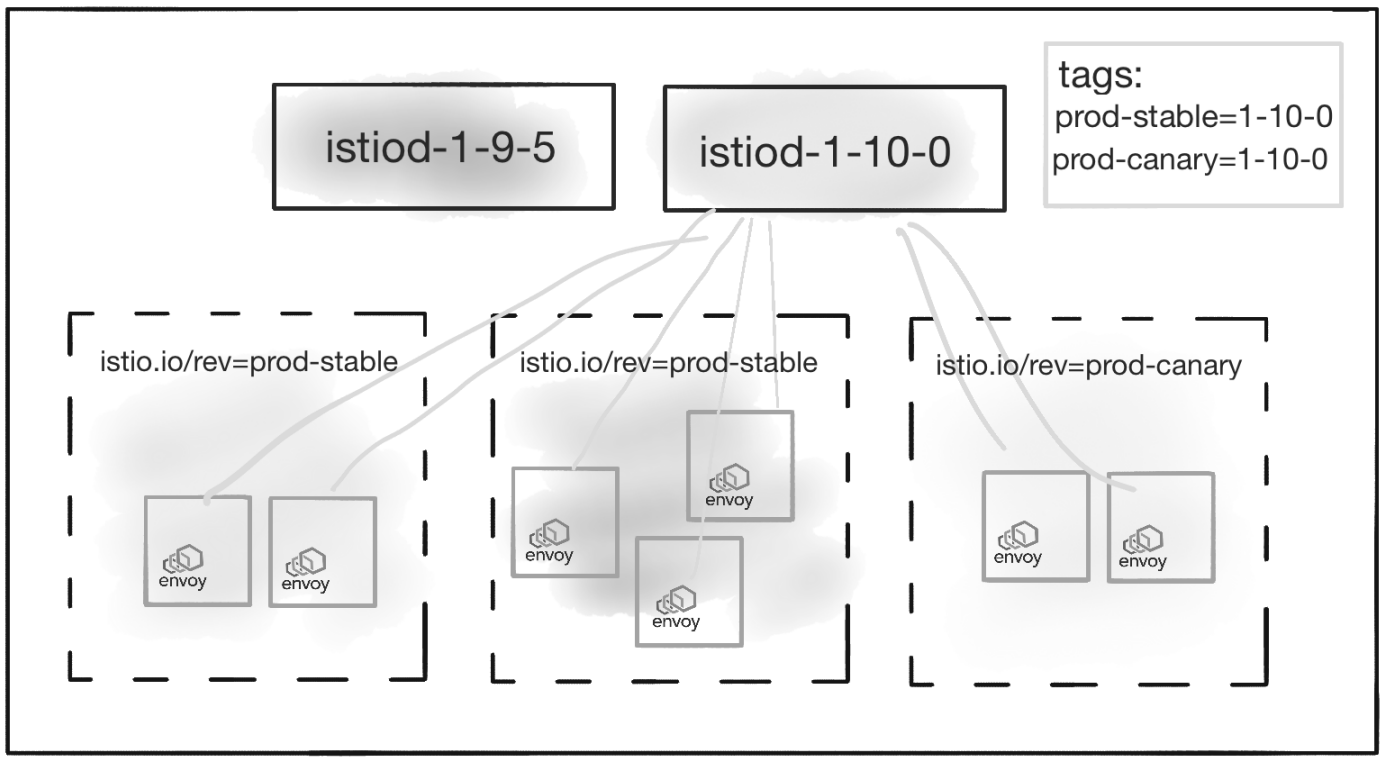
`tag list` command:

```
$ istioctl tag list
TAG         REVISION NAMESPACES
prod-canary 1-10-0   ...
prod-stable 1-9-5    ...
```

After the cluster operator is satisfied with the stability of the control plane tagged with `prod-canary`, namespaces labeled `istio.io/rev=prod-stable` can be updated with one action by modifying the `prod-stable` revision tag to point to the newer `1-10-0` revision.

```
$ helm template istiod manifests/charts/istio-control/istio-discovery -s templates/revision-tags.yaml --set revisionTags={prod-stable} --set revision=1-10-0 -n istio-system | kubectl apply -f -
```

Now, the situation is as below:



Namespace labels unchanged but now all namespaces pointed to 1-10-0

Restarting injected workloads in the namespaces marked `prod-stable` will now result in those workloads using the `1-10-0` control plane. Notice that no namespace relabeling was required to migrate workloads to the new revision.

Default tag

The revision pointed to by the tag `default` is considered the ***default revision*** and has additional semantic meaning.

The `default` revision will inject sidecars for the `istio-injection=enabled` namespace selector and `sidecar.istio.io/inject=true` object selector in addition to the `istio.io/rev=default` selectors. This makes it possible to migrate from using non-revisioned Istio to using a revision entirely without relabeling namespaces. To make a revision `1-10-0` the default, run:

```
$ helm template istiod manifests/charts/istio-control/istio-discovery -s templates/revision-tags.yaml --set revisionTags={default} --set revision=1-10-0 -n istio-system | kubectl apply -f -
```

When using the `default` tag alongside an existing non-revisioned Istio installation it is recommended to remove the old `MutatingWebhookConfiguration` (typically called `istio-sidecar-injector`) to avoid having both the older and newer control planes attempt injection.

In place upgrade

You can perform an in place upgrade of Istio in your

cluster using the Helm upgrade workflow.

This upgrade path is only supported from Istio version 1.8 and above.

Add your override values file or custom options to the commands below to preserve your custom configuration during Helm upgrades.

1. Upgrade the Kubernetes custom resource definitions (CRDs):

```
$ kubectl apply -f manifests/charts/base/crds
```

2. Upgrade the Istio base chart:

```
$ helm upgrade istio-base manifests/charts/base -n istio-sy
stem --skip-crds
```

3. Upgrade the Istio discovery chart:

```
$ helm upgrade istiod manifests/charts/istio-control/istio-
discovery \
    -n istio-system
```

4. (Optional) Upgrade the Istio ingress or egress gateway charts if installed in your cluster:

```
$ helm upgrade istio-ingress manifests/charts/gateways/isti
o-ingress \
    -n istio-system
$ helm upgrade istio-egress manifests/charts/gateways/istio
-egress \
    -n istio-system
```

Uninstall

Please refer to the uninstall section in our Helm install guide.